Rešitev

Uvozimo `numpy` in preberemo podatke.

```
import numpy as np
```

```
state = np.array([list(v.strip()) for v in open("example.txt")])
```

Vsako vrstico smo odstripali znaka za novo vrsto in jo spremenili v seznam. Stanje je zdaj

```
state
```

```
array([['v', '.', '.', '.', '>', '>', '.', 'v', 'v', '>'],
       ['.', 'v', 'v', '>', '>', '.', 'v', 'v', '.', '.'],
       ['>', '>', '.', '>', 'v', '>', '.', '.', '.', 'v'],
       ['>', '>', 'v', '>', '>', '.', '>', '.', 'v', '.'],
       ['v', '>', 'v', '.', 'v', 'v', '.', 'v', '.', '.'],
       ['>', '.', '>', '>', '.', '.', 'v', '.', '.', '.'],
       ['.', 'v', 'v', '.', '.', '>', '.', '>', 'v', '.'],
       ['v', '.', 'v', '.', '.', '>', '>', 'v', '.', 'v'],
       ['.', '.', '.', '.', 'v', '.', '.', 'v', '.', '>']], dtype='<U1')
```

Višino in širino si bomo za preglednejši rabo shranili v spremenljivki.

```
h, w = state.shape
```

Oglejmo si tiste, ki hočejo dol. Kje so?

```
y, x = np.nonzero(state == "v")
```

```
x
```

```
array([0, 7, 8, 1, 2, 6, 7, 4, 9, 2, 8, 0, 2, 4, 5, 7, 6, 1, 2, 8, 0, 2,
       7, 9, 4, 7])
```

```
y
```

```
array([0, 0, 0, 1, 1, 1, 1, 2, 2, 3, 3, 4, 4, 4, 4, 4, 5, 6, 6, 6, 7, 7,
       7, 7, 8, 8])
```

Kakšne so koordinate polj pod njimi od njih?

```
newx, newy = x, (y + 1) % h
```

`newy` v resnici potrebujemo, `newx`, ki je kar enak `x`, pa nam bo prišel prav kasneje.

Katera izmed teh polj, na katera bi šli tile, so prosta?

```
can_move = state[newy, newx] == "."
```

```
can_move
```

```
array([ True, False,  True, False,  True,  True,  True, False,  True,
       False,  True, False, False,  True,  True,  True,  True,  True,
       False,  True,  True,  True, False, False, False, False])
```

Še enkrat izpišimo stanje (da ga bomo lažje primerjali), potem premaknimo tiste, ki jih lahko in potem ponovno izpišimo stanje.

```
state
```

```
array([['v', '.', '.', '.', '>', '>', '.', 'v', 'v', '>'],
       ['.', 'v', 'v', '>', '>', '.', 'v', 'v', '.', '.'],
       ['>', '>', '.', '>', 'v', '>', '.', '.', '.', 'v'],
       ['>', '>', 'v', '>', '>', '.', '>', '.', 'v', '.'],
       ['v', '>', 'v', '.', 'v', 'v', '.', 'v', '.', '.'],
       ['>', '.', '>', '>', '.', '.', 'v', '.', '.', '.'],
       ['.', 'v', 'v', '.', '.', '>', '.', '>', 'v', '.'],
       ['v', '.', 'v', '.', '.', '>', '>', 'v', '.', 'v'],
       ['.', '.', '.', '.', 'v', '.', '.', 'v', '.', '>']], dtype='<U1')
```

```
state[y[can_move], x[can_move]] = "."
state[newy[can_move], newx[can_move]] = "v"
```

```
state
```

```
array([['.', '.', '.', '.', '>', '>', '.', 'v', '.', '>'],
       ['v', 'v', '.', '>', '>', '.', '.', '.', 'v', '.'],
       ['>', '>', 'v', '>', 'v', '>', 'v', 'v', '.', '.'],
       ['>', '>', 'v', '>', '>', '.', '>', '.', '.', 'v'],
       ['v', '>', 'v', '.', '.', '.', '.', '.', 'v', '.'],
       ['>', '.', '>', '>', 'v', 'v', '.', 'v', '.', '.'],
       ['.', '.', 'v', '.', '.', '>', 'v', '>', '.', '.'],
       ['.', 'v', '.', '.', '.', '>', '>', 'v', 'v', 'v'],
       ['v', '.', 'v', '.', 'v', '.', '.', 'v', '.', '>']], dtype='<U1')
```

Deluje, ne? Zložimo vse skupaj v program, premike v desno in dol pa zložimo v zanko, saj je večina dela enaka ne glede na smer.

```
state = np.array([list(v.strip()) for v in open("input.txt")])

changes = True
step = 0
h, w = state.shape
while changes:
    changes = False
    for dir in ">v":
        y, x = np.nonzero(state == dir)
        if dir == ">":
            xnew, ynew = (x + 1) % w, y
        else:
            xnew, ynew = x, (y + 1) % h
```

```
        can_move = state[ynew, xnew] == "."
        state[y[can_move], x[can_move]] = "."
        state[ynew[can_move], xnew[can_move]] = dir
        changes = changes or np.any(can_move)
    step += 1

print(step)
```

```
534
```

Rešitev je lepa, ker vse teče lepo, vzporedno, z minimalnim številom zank, pa tudi kar razumljivo je.

S tem se naš tečaj končuje. Upam, da ste prišli čez, se veliko naučili in se ob tem tudi zabavali.

Srečno novo leto!